School of Computer Science
Dr. Ying Zhou

COMP5338: Advanced Data Models Sem. 2/2024

Neo4j Project (Individual, 20%)

14.10.2024

1 Introduction

In this assignment, you are asked to design and build a Neo4j graph model representing airline route using data set downloaded from OpenFlights Data. You are also asked to implement Cypher queries and to anlayse their performance.

2 Data Set

The data set can be downloaded as a zip file from the unit's Canvas site. It contains four CSV files: `airports.dat`, `airlines.dat`, `planes.dat`, and `routes.dat`. We keep the extension `.dat`, which is used by OpenFlights.

The `airports.dat` file has 7698 rows; each row represents an airport with the following information (the description is copied from the OpenFlights webpage):

- **Airport ID** Unique OpenFlights identifier for this airport.
- **Name** Name of airport. May or may not contain the City name.
- **City** Main city served by airport.
- **Country** Name of the country or territory where airport is located.
- **IATA** 3-letter IATA code. Null if not assigned/unknown.
- **ICAO** 4-letter ICAO code. Null if not assigned.
- **Latitude** Decimal degrees. Negative is South, positive is North.
- **Longitude** Decimal degrees. Negative is West, positive is East.
- The last five columns contains information not relevant to this project; you can ignore them while loading data into the graph.

The following is a sample entry in the `airports.dat` file showing the first eight columns only:

airports.dat

507, "London Heathrow Airport", "London","United Kingdom", "LHR", "EGLL", 51.4706, -0.461941"

The `airlines.dat` file has 6161 rows; each row represents an airlines with the following information:

- **Airline ID** Unique OpenFlights identifier for this airline.
- **Name** Name of the airline.
- **Alias** Alias of the airline. For example, All Nippon Airways is commonly known as "ANA".
- **IATA** 2-letter IATA code, if available.
- **ICAO** 3-letter ICAO code, if available.
- **Callsign** Airline callsign.
- **Country** The name of the country or territory where airport is located.
- **Active** "Y" if the airline is or has until recently been operational, "N" if it is defunct.

The following is a sample entry in the `airlines.dat` file:

airlines.dat

324, "All Nippon Airways", "ANA All Nippon Airways", "NH", "ANA", "ALL NIPPON", "Japan", "Y"

The `planes.dat` file has 246 rows; each row represents an aircraft model with the following information:

- **Name** Full name of the aircraft.
- **IATA code** Unique three-letter IATA identifier for the aircraft.
- **ICAO code** Unique four-letter ICAO identifier for the aircraft.

The following are two sample entries in the `planes.dat` file:

planes.dat

```
"Boeing 787","787",
"Boeing 787-10","78J","B78X"
```

The `routes.dat` file has 67663 rows; each row represents a flight route between two airports with the following information:

- **Airline** 2-letter (IATA) or 3-letter (ICAO) code of the airline.
- **Airline ID** Unique OpenFlights identifier for airline (same as **Airline ID** in the `airelines.dat` file).
- **Source airport** 3-letter (IATA) or 4-letter (ICAO) code of the source airport.
- **Source airport ID** Unique OpenFlights identifier for source airport (same as **Airport ID** in the `airports.dat` file)
- **Destination airport** 3-letter (IATA) or 4-letter (ICAO) code of the destination airport.
- **Destination airport ID** Unique OpenFlights identifier for destination airport (same as **Airport ID** in the `airports.dat` file).
- **Codeshare** "Y" if this flight is a codeshare (that is, not operated by Airline, but another carrier), empty otherwise.
- **Stops** Number of stops on this flight ("0" for direct)
- **Equipment** 3-letter codes for plane type(s) generally used on this flight, separated by spaces

The following show three sample entries in the `routes.dat` file:

routes.dat

```
BA,1355,SIN,3316,LHR,507„0,744 777
BA,1355,SIN,3316,MEL,3339,Y,0,744
TOM,5013,ACE,1055,BFS,465„0,320
```

The dataset was aggregated from various sources and updated at varying paces. As a result, data consistency cannot be guaranteed. For instance, the **routes.dat** file references airline and airport IDs from the **airlines.dat** and **airports.dat** files respectively. Some of these references may correspond to non-existent records. To ensure query accuracy, it is necessary to remove such obsolete routes from the dataset. In summary, your graph should contain information about 7698 airports, 6161 airlines, 246 aircrafts, and **66316** valid routes.

3 Workloads to implement

You are asked to design a graph schema and load the data according to your schema to build the graph and implement the following workloads, each using a **single** Cypher query. You will receive zero marks for any workload that utilizes multiple queries.

NW1 Identify the top five aeroplanes that are most frequently utilized on international flight routes. An international route is defined as a flight route where the source airport and the destination airport are located in different countries/territories. Your query should return the aeroplane's name and the corresponding number of international routes it is used on, sorted in descending order based on the number of international routes. A sample output row is given below.

NW1 partial output

```
"Airbus A320", 8785
```

NW2 Identify the top five airlines that operate flights in the highest number of distinct countries/territories. An airline is considered as operating in a country/territory if it has least one flight route where either the source or destination airport is located in that country/territory. Your query should return the airline name and the number of countries/territories they operate. A sample output row is given below.

NW2 partial output

```
"Air France", 126
```

NW3 Identify the top five airports that serve a greater number of international flights compared to domestic flights by calculating the difference between their international and domestic flight counts. An international flight is defined as any flight where either the source or destination airport is located in a different country/territory while a domestic flight involves both airports within the same country/territory. Your query should return the airport name, the number of international flights, the number of domestic flights and the difference between the number of international flights and the number of domestic flights. The results should be sorted by the difference between the number of international flights and the number of domestic flights. A sample output line is given below.

NW3 partial output

```
"London Heathrow Airport",991,56,935
```

NW4 Identify all connecting flights from Sydney Kingsford Smith International Airport

(IATA code: SYD) to London Heathrow Airport (IATA code: LHR) that include a stop in one intermediate (middle) city, with both flight legs operated by the same airline. Your query should output the airline's name and the sequence of cities involved in the journey, including Sydney, the connecting city, and London. The results should be sorted by the airline's name. A sample output line is given below. There is no specific requirement for the order of the city names or the output format within a single row.

NW4 partial output

```
"Virgin Atlantic Airways", ["Sydney", "Hong Kong", "London"]
```

NW5 Identify the top five airports with the highest number of flight routes and, for each of these airports, determine the top three airlines based on the number of non-code-share routes they operate through the airport. Your query should return the name of the airport, the total number of routes operated by the airport, the name of the airline, and the number of non-code-share routes they operate through the airport. A sample output line is given below. Your workload's output format does not need to be exactly the same as the sample output format.

NW5 partial output

```
"London Heathrow Airport", 1047, [["British Airways", 256], ["Virgin Atlantic
Airways", 40], ["Finnair", 38]]
```

4 Query Design

In this section, you are asked to design and implement two queries with the following features. The queries must be based on the same graph. They must be different to queries in the previous section. The queries should be meaningful in the domain; that is, you can clearly describe the purpose of the query in the domain.

DW1 A query using at least one of the following predicate functions: ANY, SINGLE, ALL, NONE

DW2 A query using at least a spatial function and a list function.

5 Neo4j Browser Guide

All queries as described in section 3 and 4 as well as the graph setup queries should be submitted as a custom browser guide. A browser guide consists of a number of slides. Most slides should contain one or more executable Cypher queries and a textual description. It is written as a partial HTML document using the basic structure described in the Creating remote browser guides in HTML document.

You can use any text editor, including visual studio code, to open a browser guide and inspect its contents.

Your guide should have a similar structure. It must contain the following slides:

- **Slide one**: Graph Building. In this slide, you should provide one or more queries to build a graph by loading the data set. You should create necessary indexes to ensure that your graph loading takes less than a couple of minutes.
- **Slide two**: Graph Schema inspection. In this slide, provide a brief description or a diagram of the graph schema including all node labels and the relationship types. You can include the runnable command `CALL db.schema.visualization` in the slide.
- **Slide three - seven**: Include your implementation of workloads NW1 to NW5. Include the workload number and a brief description, which could be copied from the assignment instruction on each slide.
- **Slide eight**: Predicate function. Include your implementation of workload DW1 as a clickable query in this slide. You must also have a textual description part; include a brief description of the purpose of your query and which predicate function is used to implement the query.
- **Slide nine**: Spatial and list function. Include your implementation of workload DW2 as a clickable query in this slide. You must also have a textual description part; include a brief description of the purpose of your query. Highlight the spatial and the list funtions used.
- **Slide ten**: Clear the Graph. In this slide, write a query to delete all nodes and their relationships in the graph.

6 Report

You are asked to prepare a report to describe and justify your graph model. You also need to give a brief description of each query and provide a performance observation of at least three queries.

The report should have the following sections:

- Introduction
- Graph Model

- Workload Implementation
- Query Design and Implementation
- Performance Observation
- Conclusion

The introduction and conclusion sections should be very brief. There is no specific guidelines on what should or should not be put in those two sections, but they should integrate well with the rest of the report and contribute to the overall professionalism of the report.

Below are what you should put in the four middle sections:

- Graph Model. This section must include a brief description of the node labels and relationship types of your graph. You must also include a screenshot of graph schema produced by "CALL db.schema.visualization". You should also justify the design and mention alternatives you have considered in this section.
- Workload Implementation. Describe the implementations of workloads NW1 to NW5 in this section. Include the actual query and give a brief explanation of each section of the query.
- Query Design. Describe the design of query DW1 and DW2 in this section. Include the actual query and give a brief explanation of the purpose and how the required function is used in this query.
- Performance Observation. Describe the execution plan of at least three workloads among NW1 to NW5. Highlight the index usage if applicable. Describe any potential improvement or improvement you have implemented in the query.

7 Deliverable and Submission Guidelines

Both the browser guide and the report are due on Friday 23:59, 1^{st} of November, 2024. There are separate submission links for the two files.

8 Generative AI Usage Guidelines

You are permitted to use Generative Artificial Intelligence (AI) for the following:

- Obtaining suggestions on Neo4j functions that may be useful for particular classes of problems;
- Spell checking and grammar checking of text that you wrote yourself;

- Obtaining recommended readability improvements to text that you wrote yourself; and

- Obtaining translations of individual words or short phrases (but not entire sentences).

The use of such tools must be appropriately acknowledged. You can do this by including an acknowledgment section at the end of your report where you need to describe the AI tool(s) that you used, what you used it to do, what prompt(s) you provided, and how AI output was used or adapted by you.